静止画像の可逆圧縮･高速転送に関する試行

加治佐　清光†

Feasibility Study of Lossless Compression and Fast Transfer of Still Images

Kiyomitsu KAJISA

For the purpose of applications of remote product tests and remote product monitoring, to verify whether the lossless image compression is useful or not from the view point of the speed of continuous transfer of still images, using a non-expensive USB camera, we implemented test programs and tested. Results, using PNG and TIFF images by standard lossless image compression, show that overhead caused by the lossless compression and decompression introduces approximately 1/4 transfer delay compared with the case of no image compression, though the transfer speed of the compressed images themselves is shorten. This technical paper reports the method of capturing still images asynchronously using a USB camera and transferring images via 100Mbps LAN asynchronously, implementation of an image capture client test program and an image display server test program, experiments and results of using the test programs, and considerations of the experiments.

Keywords : Lossless Compression, Still Image, USB Camera, TCP/IP, C Sharp

1　まえがき

安価なUSBカメラを用いて静止画像のLAN経由連続転送の実験プログラムを試作した．本研究報告ではその概要と実験結果について報告する．

本研究の主な目的は，画質劣化のない（無歪みの）画像による遠隔製品検査や遠隔監視等の応用のために，動画像ではなく連続して静止画像を LAN 経由で転送する場合に，転送速度の観点から可逆（無歪）圧縮が有効であるかどうかを検証することである．

著者らは，平成 13 年度校内研究助成金（一般研究と教育改善プロジェクト）により整備した工業用 CCD カメラ（約 15 万円）と日立高速画像処理ボード（約 36 万円）等で構成される画像処理装置（**図 1** を参照）を使用して，平成 14 年度の卒業研究 [1] で Visual C を用いて静止画像の LAN 経由転送の実験を行い，工業用 CCD カメラから連続して取得される 512×440 画素の濃淡静止画像（8 ビット/画素）を約 5 [Frame/Sec] の転送速度で転送できることを確認した．使用した日立高速画像処理ボードには大容量の画像メモリが内蔵されており，カメラから取得した画像を内蔵の画像処理プロセッサの機能を用いて高速に BMP あるいは YUV 形式のファイルに保存可能である．

図 1　日立高速画像処理ボード等を用いた画像処理装置

しかし，静止画像転送の実験装置としては，この工業用 CCD カメラと高速画像処理ボードは高価である点と，卒業研究の時間には画像処理教材の開発でも使用するため装置の共用が困難な点から，安価なビデオカメラとビデオキャプチャボードの構成が考えられたが, Windows 用ドライバのインタフェース情報が入手困難であったため，今回は，近年普及している USB カメラ，PC カメラや Web カメラなどと呼ばれる安価な USB インタフェースの CCD カメラを使用して実験を行った．

†情報工学科

図 2　USB カメラの例

なお, 実験プログラムの試作には, Microsoft Visual Studio.NET [2] の Visual C#（シー・シャープ）を用いた．Visual C#は Visual C++と Visual Basic のオブジェクト指向と視覚化の利点を採り入れた新たな言語である．

以下, 静止画像の取得, LAN 経由転送, 試作プログラムおよび実験と考察について報告する．

2　静止画像の取得

2.1 USB カメラ

USB カメラ，PC カメラや Web カメラなどと呼ばれる安価な USB インタフェースの CCD カメラの例を**図 2** に示す. 画素数は 35 万あるいは 30 万画素であり，160×120，320×240 あるいは 640×480 画素の動画像あるいは静止画像が取得できる．今回の実験では，**図 2** 中で左端の USB カメラ [3]を 320×240 画素で使用した．

2.2 画像取得

USB カメラからの動画像と静止画像の取得は Microsoft の DirectX 中の DirectShow を使用した．DirectX は 3D グラフィックスなどのマルチメディア機能を高性能で高速に実現するための低レベルの API であり，DirectShow はビデオとオーディオのマルチメディアファイルを生成あるいは再生するための低レベルの API である．実験で使用した Windows XP パソコンには DirectX8.1 が組み込まれている．DirectX9.0b も Microsoft のサイトからダウンロードでき, 最近発売されている Windows XP パソコンには DirectX9.0b が組み込まれているが, DirectShow の言語サポートは VB と C++のみで, C#からは直接使用できない [4] . C#からメソッド等を利用するにはカプセル化する必要があるため，今回は，ネット上（The Code Project）で公開されている DShowNET.dll (52KB) [5] を使用した．

表 1　DirectShow のインタフェース，クラス，メソッド

(a) interface DShowNET.

IBaseFilter	ビデオ装置のフィルタ制御用
IGraphBuilder	フィルタグラフ構築用
ICaptureGraphBuilder2	キャプチャフィルタグラフ構築
ISampleGrabber	フィルタグラフからの取得用
IMediaControl	フィルタグラフ中のデータ制御
IVideoWindow	ビデオ再生窓の制御用
IMediaEventEx	イベント通知の制御用

(b) class DShowNET.

VideoInfoHeader	MediaType
Device	AMMediaType
PinCategory	

(c) メソッド DShowNET.

ICaptureGraphBuilder2.SetFiltergraph()	
IGraphBuilder.AddFilter()	
ICaptureGraphBuilder2.RenderStream()	接続，描写
ISampleGrabber.GetConnectedMediaType()	
ISampleGrabber.SetBufferSamples()	
ISampleGrabber.SetOneShot()	
ISampleGrabber.SetCallback()	静止画像の取得
ISampleGrabberCB.BufferCB()	静止画像の取得
IVideoWindow.SetWindowPosition()	
IMediaEventEx.GetEvent()	
IMediaEventEx.FreeEventParams()	

従来，カメラからの画像取得には VFW (Video For Windows)が使用されているが，C#による使用が困難であったためと，VFW は DirectX に移行するとのことで，今回，VFW は使用しなかった．

USB カメラの選択, 動画像の表示, 静止画像の取得に使用した DirectShow のインタフェース，クラス，メソッドの一覧を**表 1** に示す．

DirectShow を使用して USB カメラから静止画像を取得するには，まず，使用するクラスやインタフェースを宣言し, 選択したカメラ装置との接続を確立する．次に，カメラ装置のセットアップ，具体的には，選択したキャプチャ装置の生成，インタフェースの取得，キャプチャグラフのビルドを行い，得られるプレビュー用のストリームを表示ウィンドウへ描写(render)することにより動画像が表示される．

静止画像の取得は，SetCallBack( )メソッドを利用し，キャプチャのイベント発生で得られた RGB データから Bitmap を生成し，表示する．

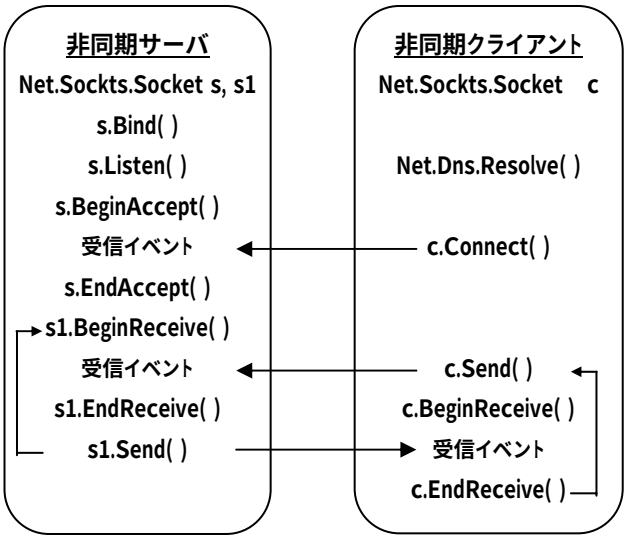

図 3　ソケットによる非同期通信 [6), 7)]

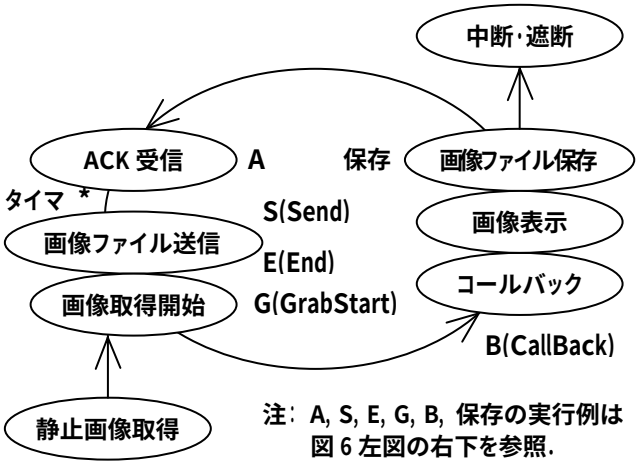

図 4　静止画像取得と転送の非同期処理（クライアント）

3　LAN 経由転送

3.1　画像の転送

平成 14 年度の卒業研究 [1)] は, TCP/IP の同期通信の手法を用いて 100Mbps と 1Gbps の LAN 経由で静止画像転送の実験を行った．しかし，同期通信においては受信開始から実際の受信までは処理が占有され，他のプログラム（スレッドやプロセス）は動作することができない欠点がある．そのため，平成 15 年度の卒業研究 [6)] では，受信はイベント（割り込み）として発生する非同期受信 [7)] の手法（**図 3** を参照）により，あらかじめ保存された標準画像を用いて，転送の実験を試みた．

通信の常套手段であるハンドシェークを用いた場合，送信側が画像を転送し，相手側が画像を受信し，画像を表示した後に受信確認の返答(ACK，Acknowledge)を返信してくるまでには，明らかな時間の差がある．この待ちの間に，非同期受信を用いれば，他の処理，ここでは静止画像の取得と画面表示を行うことができる．つまり，非同期受信を用いれば同期通信よりも高速に画像を転送できることになる．

3.2　非同期処理

しかし，試作プログラムを用いた実験の結果，非同期のデータ(ACK)受信ルーティンの中で，DirectShow を用いた静止画像取得の非同期コールバックの設定（静止画像取得開始）はできない（つまり，非同期処理の中で非同期処理は並行して実行できない）ことが判明した．そのため，10ms のタイマーを使用し，ACK 受信後に非同期コールバックを設定する（つまり，二つの非同期処理は分離する）こととした．

以上の考えに基づく，保存された画像の転送(Send～End)から応答(ACK)受信までの通信に対する非同期処理と，静止画像の取得開始(GrabStart)から静止画像取得(Call Back)・画像表示・画像ファイル保存までのカメラに対する非同期処理を一つのシーケンスとしたループを**図 4** に示す．

4　試作プログラム

4.1　画像取得クライアント

画像取得・送信を行う非同期クライアントの試作プログラム上のツールバー・ボタンを**図 5** に，動作中の例を**図 6** の左図に示す．カメラ選択ボタンに引き続く撮像ボタンにより**図 6** の左図右画面に動画像が表示される．さらに，静止ボタンにより連続して取得された静止画像が左画面に表示され，指定の画像形式（書式）で一つのファイルに連続して重ね書きで保存される．

あらかじめ, LAN 設定ボタンによりサーバと接続されていれば，ファイル保存の直後に取得した静止画像のファイルが連続してサーバへ送信される. その後は，**図 4** に示したシーケンスで連続して画像取得と画像送信が繰り返される．

送信のためにファイル保存するカラー画像の形式は，C#の Draw.Image.Save メソッドでサポートされている BMP（非圧縮）, TIFF, GIF, PNG（無歪圧縮）, JPEG

図 5　ツールバーのボタン

表 2　試作プログラムで使用した画像形式の特徴 [8)~10)]

BMP	無歪, 固定サイズ 24 bits/pixel 等
Tag Image File Format	無歪圧縮, LZW 圧縮など Tag で圧縮法等を指定 24 bits/pixel 等 グレースケールが基本
Graphics Interchange Format	無歪圧縮, LZW 圧縮 8 bits/pixel, 256 色 透明, アニメーション
Portable Network Graphics	無歪圧縮, LZ77+2 次元 Huffman 24 bits/pixel 等 透明, 逐次, γ・色補正
Joint Photographic Experts Group	有歪圧縮 24 bits/pixel 等 圧縮 1/20 で識別難 線図, 色ブロック, 境界に弱い

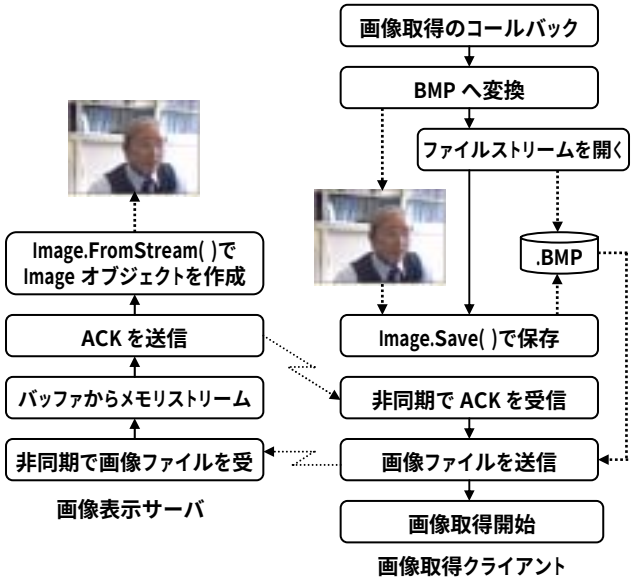

図 7　画像取得クライアントと画像表示サーバの処理の流れ

（有歪圧縮）を使用した．但し，GIF の場合は撮像したフルカラー画像が 256 色へ減色される．それぞれの画像形式の特徴については**表 2** を参照されたい．

図 4 に示した非同期静止画像取得と非同期転送の二つの非同期処理に基づく画像取得クライアントの処理の流れを**図 7** の右側に示す．

画像の取得は，具体的には，DirectShow によりコールバックされた画像データをビットマップデータへ変換し，画面表示する．その後，一つのファイル・ストリーム(stream)をあるファイル名で開き，そのファイル名を使用して設定された画像形式で画面 Image オブジェクトを Image.Save メソッドにより保存する．ソケットを使用した画像ファイルの送信はファイル名の指定だけで行う．

4.2 画像表示サーバ

画像受信・表示を行う非同期サーバの試作プログラムの動作例を**図 6** の右図に示す．また，画像表示サーバの処理の流れを**図 7** の左側に示す．非同期サーバは非同期で画像ファイルを受信直後に，固定サイズ（230,454 バイト）の BMP の場合は，ACK (Acknowledge)を送信する．ここで，受信はソケットのバッファからメモリ･ストリームへの書込みを指す. その後，ストリームから System.Drawing.Image オブジェクトを作成し，作成した Image を画面へ表示する．画像形式はストリームからビットマップ・オブジェクトの Image を作成するときに自動認識され，圧縮画像の場合は画像形式に基づき復元される．

図 6　非同期クライアント（左図）と
非同期サーバ（上図）

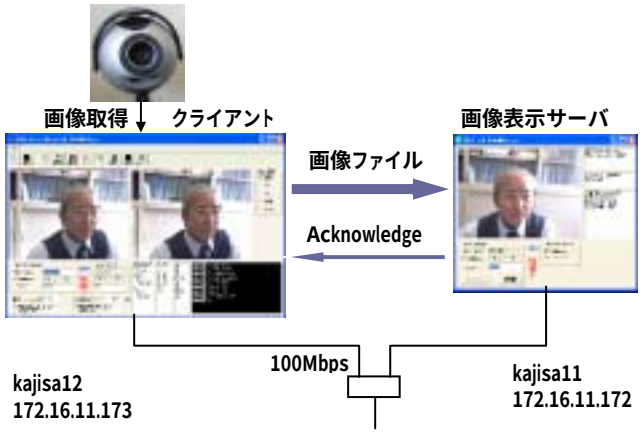

図 8　LAN 経由連続画像転送

5　実験と考察

5.1 実験環境

試作プログラムを用いた実験は Windows XP Home Edition 搭載のデスクトップパソコン(Pentium 4, 2.4GHz)とノートパソコン(Pentium 4, 2GHz)を用い，常駐プログラムや他のアプリケーションが実行されていない状態で行った．静止画像の画像取得だけの速度[Frame/Sec]と画像取得しながら転送する画像転送速度[Frame/Sec]は，約 1000 フレームの転送の時点で記録した試作プログラム（クライアント）上の表示結果（平均）を実験結果として使用した．

5.2 パソコン 1 台使用の実験

パソコン 1 台上で画像取得クライアントと画像表示サーバを試験動作させた場合の実験結果を**表 3** に示す．画像取得からファイル保存までの処理速度に関し，BMP の場合は 34.1[Frame/Sec]と余裕があり，JPEG の場合は 30.0[Frame/Sec]と動画像の画像取得速度に追従できる．しかし，GIF, TIFF, PNG の順に，指定画像形式へ無歪圧縮してファイル保存するために，画像取得速度は遅くなる．

平均画像転送速度に関しては，JPEG と GIF はファイルサイズが非常に小さいながら，BMP に比べてわずかに速いだけである．これは，クライアントでの画像圧縮に加えて，サーバでの画像復元に処理時間を要するためと，BMP の場合は画像復元･表示の前に ACK を送信できるためであると思われる．

5.3 LAN 100Mbps 使用の実験

100Mbps のハブ 1 台に接続された 2 台のパソコンを使用した場合（**図 8** を参照）の実験結果を**表 4** と**表 5** に示す．**表 4** 中のファイルサイズは**図 6** の左図に示す画像を取得した場合を示す．

表 3　パソコン 1 台で実験の場合

画像形式	画像転送 [F/s]	画像取得 [F/s]	サイズ [Bytes]
BMP	10.2	34.1	230,454
TIFF *	7.2	18.5	202,893
GIF *	10.5	25.1	36,425
PNG	7.8	16.8	140,058
JPEG	10.9	30.0	12,366

クライアント, サーバ：Pentium4 2GHz；*：LZW 圧縮

表 4　LAN 100Mbps の場合

画像形式	画像転送 [F/s]	画像取得 [F/s]	サイズ [Bytes]
BMP	21.3	46.9	230,454
TIFF	16.0	29.4	212,420
GIF	22.0	38.4	42,720
PNG	16.5	27.1	144,200
JPEG	27.9	44.0	12,560

クライアント：Pentium4 2.4GHz，サーバ：Pentium4 2GHz

表 5　LAN 100Mbps の場合

画像形式	画像転送 [F/s]	画像取得 [F/s]	サイズ [Bytes]
BMP	13.7	32.4	230,454
TIFF	10.5	20.4	203,700
GIF	15.2	21.2	39,790
PNG	11.7	19.2	134,600
JPEG	17.1	32.7	11,210

クライアント：Pentium4 2GHz，サーバ：Pentium4 2.4GHz

表 4 中の画像取得からファイル保存までの処理速度に関しては，画像形式ごとに**表 3** の実験結果と同様な傾向が見られ，全ての画像形式においてほぼ動画像取得の速度（30[Frame/Sec]）に追従できる．平均画像転送速度に関しては，JPEG の場合はほぼ動画像と同等, BMP の場合は動画像の約 2/3 のフレーム数でほぼスムーズに動画を再生できることがわかる. また, GIF は BMP と大差ないため使用する効果が低いこと，BMP に比べ TIFF は 75%に, PNG は 77%にダウンすることがわかる．特に，PNG の場合，ファイルサイズは BMP に比べて 63%と小さく無歪圧縮されるが，LAN 転送速度の向上よりも圧縮処理に要するオーバヘッドが悪影響しているものと思われる．

表 5 は画像取得クライアントを遅い方のパソコンと

した場合の実験結果であるが，**表 4** に比べて処理速度の低下が歴然となる．したがって，画像受信・表示のサーバ側よりも，画像取得・送信のクライアント側でより高速な処理が必要となる理由が裏付けられる．

6 むすび

以上，安価な USB カメラ用の試作プログラムと静止画像の LAN 経由転送の実験結果について報告した．

結果として，試みた非同期受信と画像取得の非同期処理との並行処理は静止画像の高速転送に有効であるが，C#で提供される Draw.Image.Save メソッドによる無歪画像圧縮とストリームのビットマップ・オブジェクト Image への割付けによる無歪画像復元は約 1/4 の転送遅延をもたらすことが判明した．

この結果より，近年登場したネットワーク向けの無歪画像圧縮形式である PNG に比べ，圧縮ファイルサイズより圧縮処理時間，つまり，より簡便(low complexity)で高速な圧縮処理を優先した新たな無歪画像圧縮方式を開発研究することが，無歪静止画像を連続して高速転送するために意義があると思われる．

謝辞

本研究は平成 14 年度校内研究助成金（一般研究）を受けて行われたことを記して謝意を表します．

参考文献

1) 福原輝，井上隆司，星原大翼：“ 無歪静止画像の LAN 経由転送の実験　―背景とソケット通信，画像取得と画像変換，画像表示と評価― ”，平成 14 年度卒業研究論文，鹿児島高専情報工学科，2003.

2) Microsoft Visual Studio.NET Professional Edition, Version 2002.

3) Qcam Pro 4000 取扱説明書，（株）ロジクール，2003.

4) 塚田浩二：“ USB カメラを C#で使おう ”， http://mobiquitous.com/~tsuka/programming/usbcamera.html, 2003.

5) NETMaster: “ DirectShow.NET ”, http://www.codeproject.com/cs/media/directshownet.asp, 2003.

6) 原口奈津起，勝目美由紀：“ 無歪静止画像の LAN 経由転送の実験 ―C#とソケット通信，画像表示と画像転送― ”, 平成 15 年度卒業研究論文, 鹿児島高専情報工学科, 2004.

7) “ 非同期クライアント・サーバソケット ”, .NET Framework 開発者ガイド，Microsoft Visual Studio.NET, 2003.

8) D.C. Kay and J.R. Levine 著，MbCD 訳：グラフィックフォーマット・ハンドブック，(株)アスキー，1995.

9) J. Miano: Compressed Image File Formats, Addison Wesley Longman, Inc., 1999.

10) G. Roelofs: PNG The Definitive Guide, O'Reilly & Associates, Inc., 1999.